EYES: A Novel Overtaking Assistance System for Vehicular Networks

Subhadeep Patra(✉), Javier H. Arnanz, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni

Department of Computer Engineering, Universitat Politècnica de València, Camino de Vera S/N, 46022 Valencia, Spain
subpat@doctor.upv.es, javiherar@yahoo.com, {calafate,jucano,pmanzoni}@disca.upv.es

Abstract. Developments in the ITS area are received with great expectation by both consumers and industry. Despite their huge potential benefits, ITS solutions suffer from the slow pace of adoption by manufacturers. In this paper we propose EYES, an ITS system that aims at helping drivers in overtaking. The system autonomously creates a network of the devices running EYES, and provides drivers with a video feed from the vehicle located just ahead, thus presenting a better view of any vehicles coming from the opposite direction and the road ahead. This is specially useful when the front view of the driver is blocked by large vehicles, and thus the decision whether to overtake can be taken based on the visuals provided by the application. We have validated EYES, the proposed overtaking assistance system, in both indoor and realistic scenarios involving vehicular network, and preliminary results allow being optimistic about its effectiveness and applicability.

Keywords: Android application · Real implementation · Video transmission · Live streaming · RTSP · Vehicular network · ITS

1 Introduction

Intelligent Transportation Systems (ITS) are advanced solutions that make use of vehicular and infrastructured networks to provide innovative services related to both traffic and mobility management, and that interface with other models of transport. ITS aims at using the already available transport networks in a smarter manner, resulting in significant coordination and safety improvements. Our goal here is to *integrate smartphones into vehicular networks* to develop ITS applications that can reach out to the masses in a short period of time. The choice of smartphones is not only justified by their wide availability and use, but also because they are evolving towards high performance terminals with multi-core microprocessors packed with sufficiently accurate onboard sensors.

The architecture and application developed has been named EYES. It has been developed for the Android platform, and requires the devices running it to

S. Papavassiliou and S. Ruehrup (Eds.): ADHOC-NOW 2015, LNCS 9143, pp. 375–389, 2015.
DOI:10.1007/978-3-319-19662-6_26

be equipped with at least a GPS and a back camera. The application makes use of the camera to record video and transmit it over the vehicular network, thus providing an enhanced multimedia information aid for overtaking. The location information of the vehicles gathered from the GPS is useful since the transmission of the video feed only occurs between cars travelling in the same direction, and always occurs from the vehicle in front to the vehicle travelling behind. The Android device is to be placed on the vehicle dashboard with the camera facing the windshield, so that a clear view of the road in front and cars coming from the opposite direction can be captured. Once started, the application requires no further user interaction to operate, and it can run in the background. The EYES application can be specially useful in scenarios where the view of the driver is blocked by a larger vehicle, or when a long queue of cars is located ahead and the driver wishes to overtake. In this case, the application will automatically receive the video feed from the vehicle right in front, and play the received feed on screen, thus aiding the driver in deciding the safest moment to overtake. A more detailed explanation of the architecture, design, and implementation issues of the EYES application will be provided in the following sections.

The rest of this paper is organized as follows: In Sect. 2, we survey some works in the literature that are closely related to our own. In Sect. 3, we will present an overview of the developed application. Later, in Sect. 4, we will present a description of EYES, its modules, and some implementation details. The setup used to deploy and validate EYES will be described in detail in Sect. 5. In Sect. 6, we will validate the application in a real testbed. Finally, Sect. 7 concludes this paper summarizing our contributions.

2 State of the Art

Both academia and industry have shown a strong interest in the field of ITS, resulting in the development of many innovative applications. Since our EYES application is targeted for smartphones, thus here in this section we are going to focus the bulk of our attention to some of the most interesting smartphone applications that are related to safe driving.

Most drive safety applications usually aim at warning generation based on onboard location sensors like in the works of Whipple et al. [1], Yang et al. [2], Diewald et al. [3] and Tornell et al. [4]. The application developed by Whipple et al. warns drivers when driving at high speed near schools. Yang et al. were concentrated on finding out the probability of accidents based on the location information. DriveAssist, by Diewald et al., triggers warning messages for certain traffic incidents, while Tornell et al., in their proposed application, display on screen important vehicles like ambulances and police cars on a map view, and they later improved that same solution in [5]. Few other applications used the On Board Diagnostics (OBD-II) [6] interface to detect incidents, like in the work of Zaldivar et al. [7] which aimed at detecting accidents. Wideberg et al. [8] also made use of OBD-II devices to extract safety and environment related information.

Only very few applications concentrated on providing visual aids to the drivers, like in SignalGuru described in [9], which leverages collaborative sensing on windshield-mount smartphones, in order to predict traffic signals' future schedule. The CarSafe App [10] was introduced by You et al., for instance, analyses the images from front and back cameras of smartphones to monitor the driver as well as the road ahead. Another interesting application available for download is iOnRoad [11], which aims at providing driving assistance functions including augmented driving, collision warning and "black-box" like video recording.

Despite the fact that we have found many different drive safety applications for smartphones, only a handful aimed at providing visual aids to the drivers, namely the SignalGuru, CarSafe and iOnRoad. However, none of these smartphone based applications actually provides real-time visual overtaking aid from other cars taking advantage of vehicular networks, even though the idea of video based overtaking assistance systems is not new. Works like the See-Through System [12] which was later improved in [13], although are not targeted for smartphones, are focused on the issue of video based overtaking assistance. Other related works worth mentioning are [14] and [15], which demonstrate the feasibility of such video based assistance systems. Improvement in performance of a video based overtaking assistant on undertaking codec channel adaptation, is shown in [14]. While, [15] is focused on reallocation of wireless channel resources to enhance the visual quality.

Encouraged by the findings from the above mentioned works and in order to fill the necessity of a visual overtaking assistance application that is targeted for the consumer section, which would require no additional hardware, but just simple download and install, we decided to develop EYES. The developed application is targeted for smartphones to achieve rapid acceptance and study the integrability of smartphones to vehicular networks.

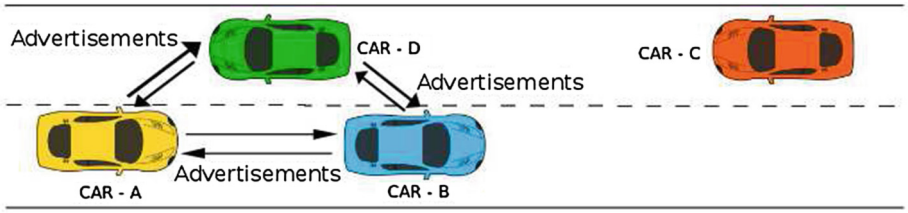
3 Overview of the EYES Architecture

The goal of the EYES application is providing assistance during overtaking by streaming real-time video coming from one vehicle to another. The minimum requirements for running EYES is the availability of a device with GPS and back camera, along with a vehicular network for the transmission of video. This section presents an overview of the functionality of the EYES architecture.

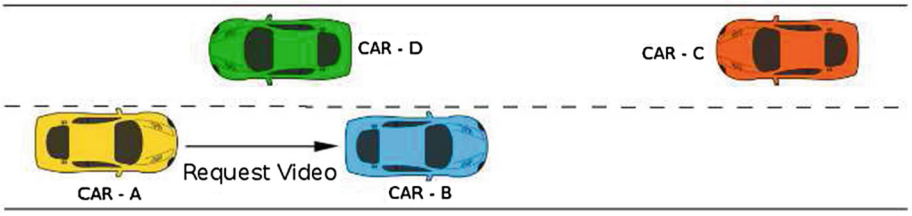
The functionality of EYES can be divided in two phases for easy understanding. *Phase one*, i.e. *Server-Client Role Establishment Phase*, involves *electing the sender and the receiver* of the video which is subject to some special tests and validation conditions. And, *phase two*, named as *Video Streaming Phase*, involving the actual *video transmission* between the sender and receiver chosen in phase one.

In the first phase, each device equipped with a back camera and running EYES *broadcasts* an advertisement containing its location and direction, while simultaneously listening for incoming broadcast messages coming from other devices. If a device receives broadcast messages from other devices, it first verifies whether the source of the message is valid. The *validity check* is based on

tests which basically involve checking if the source and destination vehicles are traveling one ahead of the other, and in the same direction. For a more detailed description of these validation conditions refer to Sect. 4.1. If several valid sources are found, the device *requests* video from the best source, which is selected based on the distance between sender and receiver devices. The source vehicle, upon receiving the request to send video from the destination vehicle, starts *streaming* the video signal over the vehicular network, in phase two. However, before sending the video, the source double-checks the validation conditions used in phase one. The destination vehicle starts playing the video onscreen as soon as it starts receiving it. The streaming and playback process is stopped only when the vehicle behind successfully overtakes, or when it stops following the vehicle in-front.



(a) The vehicles exchange *advertisements*.



(b) The client *requests* video from the server.

Fig. 1. EYES: Server-client role establishment phase.

Figure 1 provides more details about phase one. In this example, we have four cars, all of them using EYES. CAR-A and CAR-B are travelling in one direction, while CAR-C and CAR-D travel in the opposite direction. First, the cars broadcast the advertisement to each other as shown in Fig. 1a. Since CAR-C is not within the range of any other car, nobody is able to communicate with it. Each car, upon receiving the advertisement, performs the validity checks to see if the sender of the advertisement is travelling ahead in the same direction and the same lane. In this case, only CAR-A finds the advertisement message from CAR-B to be valid, and thus requests video from it, as depicted in Fig. 1b.

Similarly, Fig. 2 shows that CAR-B, upon receiving the video request from CAR-A, rechecks the validity conditions and starts streaming the video. CAR-A starts receiving the video stream and plays it onscreen immediately for its

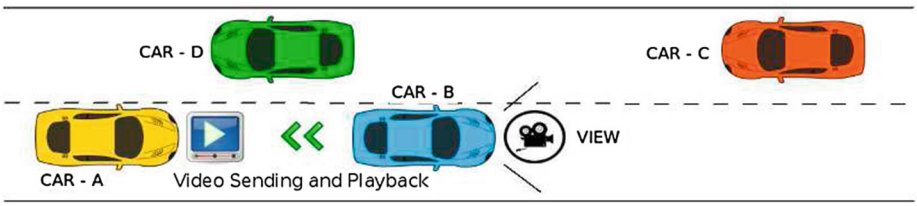


Fig. 2. EYES: Video streaming phase.

driver. It may be noted here that a single device can act both as video source and destination. This is because, while a device is receiving video from another device, it may also be streaming its own video capture to a completely different device.

4 Implementation Details of EYES

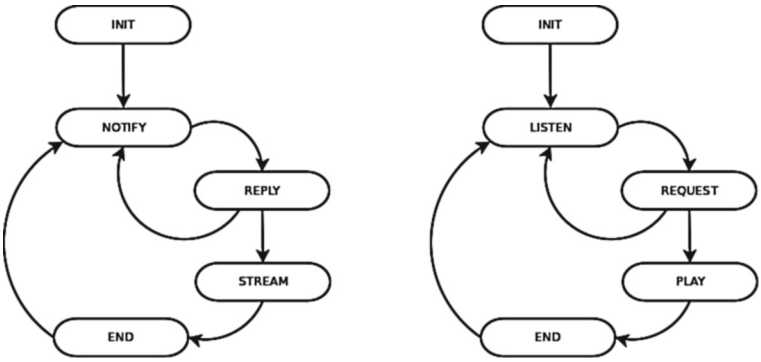
From the previous section we already know that EYES has been implemented by splitting its functionality into two distinct phases: (i) The *Server-Client Role Establishment Phase*, in which election of the video source and destination takes place; also, the starting and stopping of the video stream is controlled in this phase; (ii) The *Video Streaming Phase*, on the other hand, encompasses video streaming and playback.

The *client-server role establishment phase* is the most important of the two phases, and is run right from the start of the application until EYES is completely stopped. Even when the *video streaming phase* is transmitting video, the *client-server role establishment phase* keeps working in the background to check if overtaking has occurred and stops the video streaming. Below we describe in detail the two phases of the EYES application.

4.1 Server-Client Role Establishment Phase

This phase, as the name suggests, is in charge of choosing the server or the source of the video, and the client, i.e. the destination, for the *video streaming phase*. The starting and stopping of the video transmission will take place only under some validation conditions, namely the *same direction test* and the *same lane test* for starting the video transmission, while for ending the transmission, the *overtake test* is used. Even though, at the beginning of this phase, the server and client roles are not established, we will use the words server and client to refer to the devices that will be attaining the respective role in the future for the sake of clarity.

Figure 3 shows the different states that a server and client can attain. When the client and the server start, the server is in the *notify* state, as shown in Fig. 3a, and it starts advertising the availability of the video by broadcasting a *hello* message. Besides sending advertisements, the server, while in *notify* state,

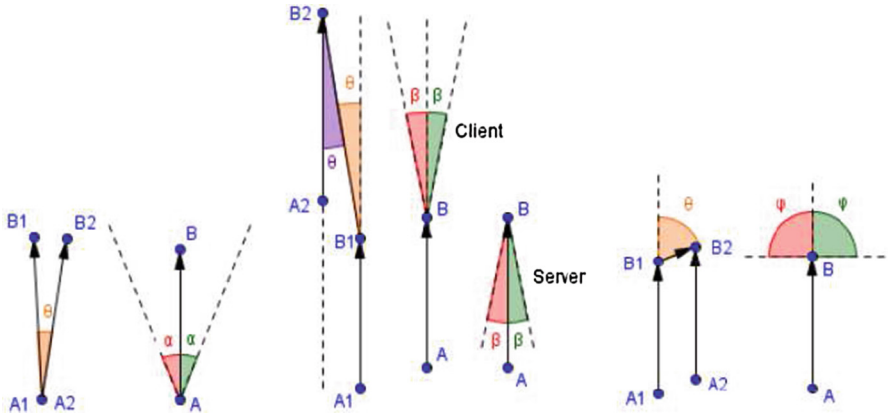


(a) Different Server States. (b) Different Client States.

Fig. 3. State diagram of the server and client.

also listens for replies to its *hello* message from clients requesting the video feed. A *hello* message contains the location information of the server so that the client, upon receiving it, can determine if the server is ahead of the client and travelling in the same direction. The client remains listening for advertisements from the server for only a certain period of time, while in the *listen* state, as shown in the Fig. 3b.

If the client receives *hello* messages from different servers, the client checks whether the servers are valid, and stores them in a queue of candidate servers. The proposed validity tests include the *same direction test* and the *same lane test* conditions as shown in Fig. 4.



(a) Same direction test. (b) Same lane test. (c) Overtake test.

Fig. 4. The different validation conditions in EYES.

The *same direction test* is used to detect whether two vehicles are travelling in the same direction. For understanding the *same direction test*, let us assume we have two cars, one travelling from a point A1 to B1 and the other from A2 to B2 as shown in Fig. 4a. Notice that, even if two cars are travelling in the same direction and speed, its it hard for them to have an overlapping displacement vector, in other words, the angle between the two vectors is not 0. This can happen due to different driving styles and GPS errors. Thus, we measure the angle θ between these two vectors and compare it to a predefined threshold α. If θ is less than α, we can safely assume that the two vehicles are travelling in the same direction. Now, even if two vehicles are travelling in the same direction, it does not necessarily mean that one is ahead of the other, both vehicles may be travelling on different lanes or parallel roads altogether. To check if one is following the other one on the same lane, we perform the *same lane test*, and for this purpose we draw an imaginary line joining the current locations of the two vehicles, as shown in Fig. 4b, where B1 and B2 are the current locations. Then we measure the angle of intersection of this line joining the points B1 and B2 with the displacement vectors of the vehicles. When the measured angle of intersection θ is less than a predefined angle β, then the vehicles are considered to be travelling on the same lane. Being on different lanes will result in a higher value of the measured angle θ, and the *same lane test* will fail. If these two conditions are satisfied, then the two vehicles are assumed to be travelling in the same direction, one following the other.

Table 1. Messages exchanged between the server and client.

Message Type	From → To	Client State	Server State	Message Contents
Hello	S → C	Listen	Notify	Location and Direction
Request	C → S	Request	Notify	Location and Direction
Ready	S → C	Request	Reply	Video sender port
Reject	S → C	Request	Reply	-
Data	S → C	Play	Stream	Location, Direction and Speed
Data-Ack	C → S	Play	Stream	-
End	C → S	Play	Stream	-

The client which was listening for server advertisements, as soon as its listening period is over, chooses the best server from the list of candidate servers based on its distance to the server. The client then tries to connect to the chosen server by sending a *request*, and moves to the *request* state. The server, upon receiving the *request* from the client, also checks its validity by performing the *same direction* and *same lane* tests once again. Before sending the *ready* or *reject* message which denotes whether it is ready to send video being captured by its camera, the server changes its state to *reply*. The server may further choose to change

its state back to *notify* or to *stream* modes depending on its own reply. The client, which was previously in the *request* state, only changes its state to *play* if the reply from the server was a *ready* message containing the *video sender port* number, otherwise it may choose to contact some other server. Table 1, details the packet types used during the *server-client role establishment phase*.

In case the server and client are in the *stream* and *play* states respectively, the *video streaming phase* is launched to transmit the video. In fact, the *server-client role establishment phase* remains active even if the *video streaming phase* has been started. The server during this period keeps sending *data* messages containing its location information, direction and speed. This way, its corresponding client can check whether an overtake has occurred, and so the client can request the server to terminate the video stream by sending an *end* message. To find out if an overtake has been successful done, the *overtake test* takes place, as shown in Fig. 4c. This test is similar to the *lane test* condition, the only difference being that the angle θ measured here is the other linear pair of the angle of intersection between the displacement vector and the line formed by joining the current location of the two vehicles. Also, the threshold φ used here is usually a much larger value.

Upon receiving the *data* message from the server, the client, if still has not overtaken as suggested by the *overtake test*, replies the server with *data-ack* to keep the connection alive. Hence, the video streaming is continued. When the video streaming has been stopped, the client switches to the *end* state and later on moves back to the *listen* state once again. The server, on the other hand, can move to the *end* state upon receipt of the *end* message from the client or if the waiting time for a *data-ack* from the client expires. This waiting time is used to detect cases of eventual disconnections.

4.2 Video Streaming Phase

In this phase, the actual video streaming and playback takes place. The *video streaming phase* starts when the client and server of *server-client role establishment phase*, are in the *play* and *stream* states respectively. The sender, which is the source of the video in this phase, is based on the libstreaming [16] API that relies on the RTSP [17] protocol for streaming video over the network. The video is encoded by the sender using the H.264 encoding format before sending. The receiver, on the other hand, is based on libvlc [18] for decoding and displaying the received video. Apart from playing the video on screen, the device at the destination also displays to the driver, the speed information of the vehicle travelling ahead. The speed information of the vehicle ahead is extracted from the *data* message received by the client as part of the *server-client role establishment phase*.

Figure 5 shows that the sender initially awaits for an incoming connection from a receiver on a predefined port, say A. This port A becomes known to the receiver device from the *ready* message send by the server in the *server-client role establishment phase*. The receiver contacts the sender by using a port X, and then the next three steps are followed: step-i is the setup of the video

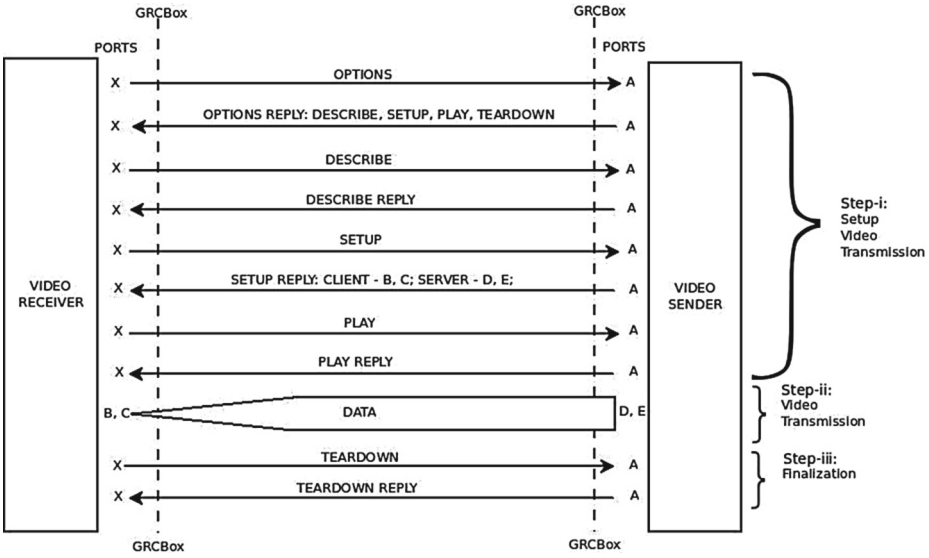


Fig. 5. The video transmission.

transmission; step-ii is when the video data transfer occurs; and step-iii involves the finalization of the video transmission. In step-i, the receiver requests the server for the supported *options*. The sender replies to the receiver with the list of supported options; in our case the sender supports *describe, setup, play and teardown*. The receiver then asks the sender to *describe* the video that it is going to send, and the sender replies. Next, the receiver requests the sender to configure the video streaming, using the *setup* option. As a result, the sender replies to the receiver with the port numbers that will be used for sending and receiving of the audio and video, which in this case are B and C for receiving, and D and E for sending. Then, the receiver tries to open the ports requested by the sender, namely B and C, and then it sends the *play* request. The sender acknowledges the play request from the receiver. In step-ii, *Data* transfer starts between the sender and receiver using the ports chosen in step-i. Notice that UDP is used in this step, whereas in the rest of the steps TCP is used. On overtaking, the receiver requests the sender to stop transmitting the video by sending the *teardown* request in step-iii. The sender acknowledges the teardown request, and the video streaming stops.

5 Deployment

For proper operation, the EYES application assumes the availability of a vehicular network, although the vehicles we use on a daily basis still lack the capability to communicate with one another.

So, for testing our application, we equipped cars with a GRCBox [19] inside them. GRCBox is a low cost connectivity device that allows the integration of

smartphones into vehicular networks and is based on the Raspberry Pi. It was developed mainly due to the difficulty in creating an adhoc network using smartphones. Another important feature that the GRCBox provides is the support for V2X communication. The different networks supported by the GRCBox include adhoc, cellular, wifi access points, etc. Thus, we use the adhoc network to create a vehicular network for EYES.

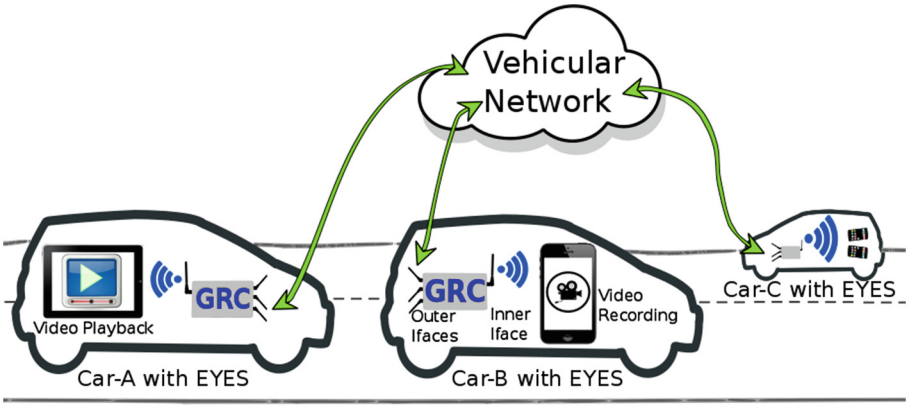


Fig. 6. Example of EYES working together with GRCBox.

Figure 6 shows how EYES works when combined with GRCBox. Each car within the vehicular network has a GRCBox mounted. The smartphones of the passengers within the car are connected to the GRCBox, and the GRCBox which is equipped with Wifi-enabled USB interfaces to communicate in adhoc mode, creates a vehicular network. Even though GRCBox is supposed to be equipped with 802.11p for vehicular communication, we used 802.11a devices instead, as 802.11p-enabled hardware was not at our disposal while setting up the GrcBox to perform the tests. For our future experiments, we intend to get hold of 802.11p compatible hardware to take advantage of the WAVE standard. As shown in the figure, Car-B is ahead of the Car-A, and both of them are travelling in the same direction and running EYES, so the smartphone in Car-B starts recording the video autonomously and sends it to Car-A, by relying on the vehicular network created using the GRCBoxes available within the cars. Concerning the video, it is played onscreen on the device in Car-A as soon as video reception starts.

We have taken advantage of the adhoc communications support of GRCBox to perform our experiments with the EYES application. The Android devices used in combination with the GRCBoxes, are a Nexus 7 and a Samsung Galaxy Note 10.1 (2014 Edition). The Nexus 7 from Google was powered by a quad-core 1.2 GHz processor, ULP GeForce GPU, 1 GB ram and 1.2 MP camera. The Samsung Galaxy Note 10.1, on the other hand, was equipped with a quad-core 1.9 GHz plus quad-core 1.3 GHz processors, 3 GB ram, 8 MP primary camera and 2 MP secondary camera.

6 Validation Test

In the EYES application, the three important conditions evaluated were described in Sect. 4.1, and each of these conditions, namely *same direction test*, *same lane test* and *overtake test*, are dependent on a threshold value. Our aim was to evaluate reasonable values of the threshold angles α, β and φ for two cars of which one follows the other throughout the experiment while travelling along a particular route, so that there is non-stop streaming of video between the cars. The cars were equipped with GRCBox devices, which helps to create a vehicular network, as described in Sect. 5. The Android devices used were the Samsung Galaxy Note 10.1 in the car ahead, and the Nexus 7 in the car that was following it.

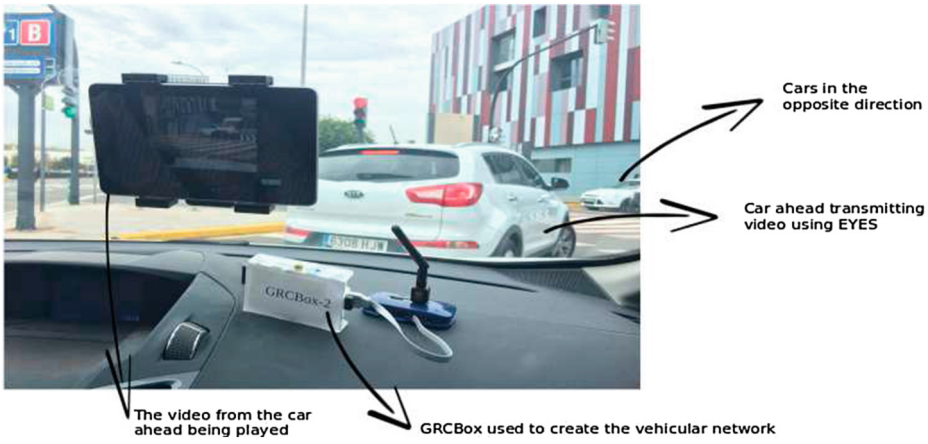
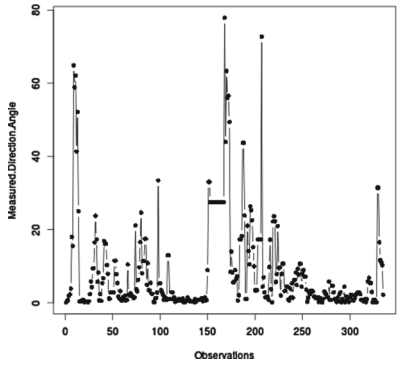


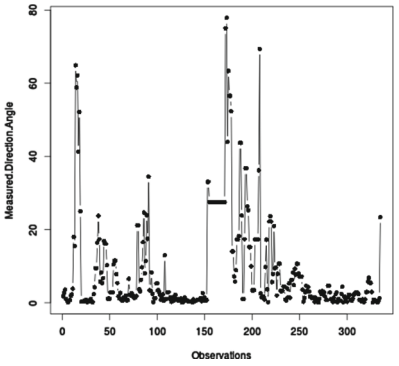
Fig. 7. The experiments with EYES in real scenario.

Figure 7 shows a photo taken during an outdoor test. In this picture, we can see that the front car is trying to take a right turn, and the back car is receiving the video from the car ahead and playing it onscreen. While doing our outdoor tests with EYES, we collected the various angles used in the three different validation tests. Below we can see the graphical representation of the data obtained during the experiment.

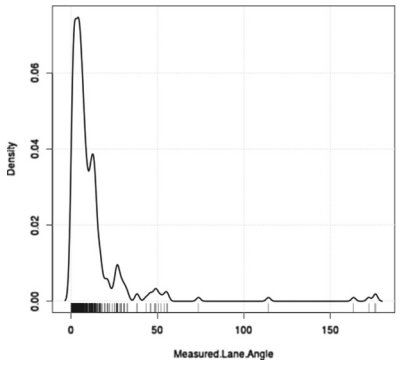
Figure 8 shows the results of the test conditions for starting the video transmission between the server and the client. Since these conditions are tested both by the server and the client, we have two sets of results. Notice that the angles shown may not have been measured at the exact same time by the server and client. Figures 8a and 8b show the plot of angles measured by the *same direction test* at the client and the server end, respectively. Most observations for both the client and server lies within 20 degrees, which is satisfactory. It is also noticeable that many peaks occur due to GPS errors, also because the route followed had



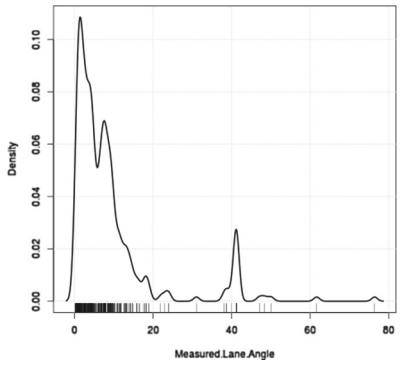
(a) Angles measured by the *same direction test* at the client side.



(b) Angles measured by the *same direction test* at the server side.



(c) Angles measured by the *same lane test* at the client side.



(d) Angles measured by the *same lane test* at the server side.

Fig. 8. Results of the *same direction* and *same lane* tests.

a lot of turns and curves, and so the two cars were not always on a straight path. Figures 8c and 8d show the density graph for the *same lane test* for the client and the server, respectively. From the two density curves, we can see that most observations for the *same lane test* lie in the range between 20–25 degrees. Notice that this value is too high considering that this test is very sensitive, and used to detect if cars are travelling on the same lane, and so we find that this condition may not be too useful when considering the accuracy of current technology.

Figure 9, shows the density plot for the observations of the *overtake test*. Note that the values used in this graph are 180 - *the observed value*, to make it more simple. Since this test is only performed by the client, we have only its data. We find that the results from this test were pretty much what we expected since all of plotted values are below 90 degrees. Furthermore, it should be noted that,

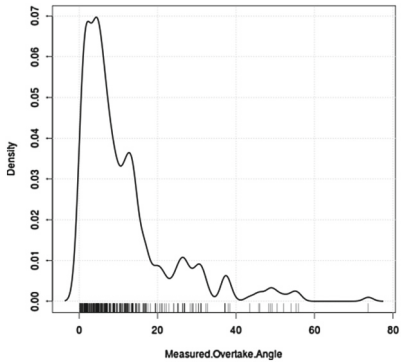


Fig. 9. Results of the *overtake test* from the client.

based on some of our indoor experiments, we have found that the application still has some minor problems related to delay of the order of few seconds between capture and playback.

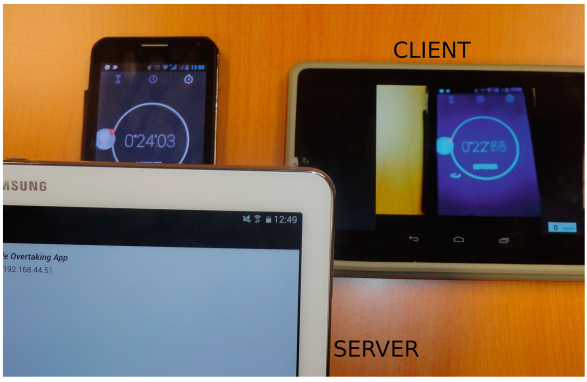


Fig. 10. The delay in one of the indoor experiments.

Figure 10 shows the delay in video recording, transmission and playback during one of our experiments carried out indoors in our laboratory. In this test, the Samsung Note 10.1 was configured to work as the server, recording the video of a stopwatch in front of it. The Nexus 7 tablet was configured to receive the video and display it. When the actual timer hit 24.03 seconds, the video being played at the client side still showed the timer to be at 22.68 seconds. Thus, there was a delay of about 1.35 seconds. Overall, from the experiments carried out, we conclude that the EYES application works correctly even though some delay issues in transmission and playback must be addressed. The validation conditions used in EYES have been validated in a real testbed, and satisfactory

results have been achieved for the *same direction test* and *overtake test*, which are used for starting and stopping the video transmission respectively. We are currently working on reducing the observed delay.

7 Conclusions

In this paper, we have presented a drive safety application called EYES that is able to help drivers in safe overtaking. The EYES system provides a real-time video feed captured by the smartphone installed in the vehicle ahead, to the smartphone of the driver seated in the car behind, which displays the video without user intervention. Thus, it provides drivers with important information and helps them to decide whether it is safe to overtake. We have evaluated the different test conditions used in EYES, and found that thresholds in between 20–25 degrees for the *same direction test* and 90 degrees for the *overtake test* are reasonable. Nevertheless, the *same lane test* was found to be useless unless more accurate GPS hardware is made available. Also, the delay of the video transmission and playback is found to be of the order of a few seconds, in some cases. Despite these minor issues, we acknowledge the fact that combining smartphones with vehicular networks indeed opens a new horizon for ITS applications and, in the future, we will focus our attention on improving our EYES application by minimizing the delay and finding an effective substitute for the *same lane* test.

Acknowledgments. This work was partially supported by the *European Commission* under *Svāgata.eu*, the Erasmus Mundus Programme, Action 2 (EMA2) and the *Ministerio de Econom?a y Competitividad, Programa Estatal de Investigaci?n, Desarrollo e Innovaci?n Orientada a los Retos de la Sociedad, Proyectos I+D+I 2014*, Spain, under Grant TEC2014-52690-R.

References

1. Whipple, J., Arensman, W., Boler, M.S.: A public safety application of gps-enabled smartphones and the android operating system. In: 2009 IEEE International Conference on Systems, Man and Cybernetics. SMC 2009, pp. 2059–2061. IEEE (2009)
2. Yang, J., Wang, J., Liu, B.: An intersection collision warning system using wi-fi smartphones in vanet. In: 2011 IEEE Global Telecommunications Conference (GLOBECOM 2011), pp. 1–5. IEEE (2011)
3. Diewald, S., Möller, A., Roalter, L., Kranz, M.: Driveassist-a v2x-based driver assistance system for android. In: Mensch & Computer Workshopband, pp. 373–380 (2012)
4. Tornell, S.M., Calafate, C.T., Cano, J-.C., Manzoni, P., Fogue, M., Martinez, F.J.: Implementing and testing a driving safety application for smartphones based on the emdr protocol. In: Wireless Days (WD), 2012 IFIP, pp. 1–3. IEEE (2012)
5. Patra, S., Tornell, S.M., Calafate, C.T., Cano, J.-C., Manzoni, P.: Messiah: an its drive safety application. In: XXV Jornadas Sarteco, Valladolid, Spain (2014)
6. International Organization for Standardization. Iso 14230–1:1999: Road vehicles, diagnostic systems, keyword protocol 2000 (1999)

7. Zaldivar, J., Calafate, C.T., Cano, J.-C., Manzoni, P.: Providing accident detection in vehicular networks through obd-ii devices and android-based smartphones. In: 2011 IEEE 36th Conference on Local Computer Networks (LCN), pp. 813–819. IEEE (2011)
8. Wideberg, J., Luque, P., Mantaras, D.: A smartphone application to extract safety and environmental related information from the obd-ii interface of a car. Int. J. Veh. Syst. Model. Test. **7**(1), 1–11 (2012)
9. Koukoumidis, E., Martonosi, M., Peh, L.-S.: Leveraging smartphone cameras for collaborative road advisories. IEEE Trans. Mob. Comput. **11**(5), 707–723 (2012)
10. You, C.-W., Lane, N.D., Chen, F., Wang, R., Chen, Z., Bao, T.J., Montes-de Oca, M., Cheng, Y., Lin, M., Torresani, L., et al.: Carsafe app: alerting drowsy and distracted drivers using dual cameras on smartphones. In: Proceeding of the 11th annual international conference on Mobile systems, applications, and services, pp. 13–26. ACM (2013)
11. ionroad official website. http://www.ionroad.com/. Accessed 8 February 2015
12. Olaverri-Monreal, C., Gomes, P., Fernandes, R., Vieira, F., Ferreira, M: The see-through system: A vanet-enabled assistant for overtaking maneuvers. In Intelligent Vehicles Symposium (IV), 2010 IEEE, pages 123–128. IEEE, 2010
13. Gomes, P., Olaverri-Monreal, C., Ferreira, M.: Making vehicles transparent through v2v video streaming. IEEE Intell. Transp. Syst. **13**(2), 930–938 (2012)
14. Vinel, A., Belyaev, E., Egiazarian, K., Koucheryavy, Y.: An overtaking assistance system based on joint beaconing and real-time video transmission. IEEE Trans. Veh. Technol. **61**(5), 2319–2329 (2012)
15. Belyaev, E., Molchanov, P., Vinel, A., Koucheryavy, Y.: The use of automotive radars in video-based overtaking assistance applications. IEEE Intell. Transp. Syst. **14**(3), 1035–1042 (2013)
16. The libstreaming android api. https://github.com/fyhertz/libstreaming. Accessed 5 February 2015
17. Schulzrinne, H.: Real time streaming protocol (rtsp) (1998)
18. Libvlc documentation. http://www.videolan.org/vlc/libvlc.html. Accessed 5 February 2015
19. Tornell, S.M., Patra, S., Calafate, C.T., Cano, J.-C., Manzoni, P.: Grcbox: Extending smartphone connectivity in vehicular networks. International Journal of Distributed Sensor Networks (2014)